

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Before the Board of Patent Appeals and Interferences

In re the Application

Inventor : **HOOGERBRUGGE**
Application No. : **10/519,649**
Filed : **12/30/2004**
For : **MULTI-PROCESSOR COMPUTER SYSTEM**

APPEAL BRIEF

On Appeal from Group Art Unit 2181

Date: 5/13/08

By: Michael Ure
Attorney for Applicant
Registration No. 33,089

Certificate of Fax/Mailing Under 37 CFR 1.8

I hereby certify that this correspondence is being faxed to (571)273-8300 or deposited with the United States Postal Service as first class mail in an envelope addressed to the COMMISSIONER FOR PATENTS, Mail Stop Appeal, P.O. BOX 1450 ALEXANDRIA, VA 22313 on date below.

Michael Ure
(Name)

(Signature and Date)

TABLE OF CONTENTS

	<u>Page</u>
I. REAL PARTY IN INTEREST.....	3
II. RELATED APPEALS AND INTERFERENCES.....	3
III. STATUS OF CLAIMS.....	3
IV. STATUS OF AMENDMENTS.....	3
V. SUMMARY OF THE CLAIMED SUBJECT MATTER ..	3
VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL	9
VII. ARGUMENT.....	10
VIII. CONCLUSION.....	11
APPENDICES: THE CLAIMS ON APPEAL.....	15

RELATED PROCEEDINGS

EVIDENCE

TABLE OF CASES

NONE

I. REAL PARTY IN INTEREST

The real party in interest is NXP B.V., the successor in interest to the present assignee of record of the present application, Koninklijke Philips Electronics N.V., and not the party named in the above caption.

II. RELATED APPEALS AND INTERFERENCES

With regard to identifying by number and filing date all other appeals or interferences known to Appellant which will directly effect or be directly affected by or have a bearing on the Board's decision in this appeal, Appellant is not aware of any such appeals or interferences.

III. STATUS OF CLAIMS

Claims 1-8 and 10 are pending, all of which stand finally rejected and form the subject matter of the present appeal. Claim 9 was erroneously omitted from the application.

IV. STATUS OF AMENDMENTS

All amendments have been entered. No amendment after final rejection has been submitted.

V. SUMMARY of the CLAIMED SUBJECT MATTER

The present invention relates to the control of the wake-up and sleep conditions of multiple processors in a multi-processor system. The claims recite in part a process list

stored accessible to the processors (e.g., stored in common memory) and a global wake-up variable loaded into cache memories of the respective processors. Changes in the global wake-up variable trigger wake up inspection of the process list, which inspection in turn determines the subsequent power state of the respective processors.

The following analysis of independent claim 1 is presented for convenience:

Element	Figure(s)	Paragraph(s) and/or page(s)
1. Multi-processor computer system comprising		
at least two processors (1) for parallel execution of processes,	Fig. 5, 1	Page 3, line 31 to page 4, line 4; page 5, lines 19-27.
at least two cache memory units (2), each being associated with and connected to a separate processor (1),	Fig. 5, 2	Page 3, line 31 to page 4, line 4; page 5, lines 19-27.
a connection bus (4) connecting said processors (1) and said cache memory units (2). and	Fig. 5, 4	Page 3, line 31 to page 4, line 4; page 5, lines 19-27.
a process list unit (3) connected to said connection line (4) for storing a process list of processes to be available for execution by said processors (1), wherein said processors (1) are adapted for	Fig. 5, 3	Page 4, line 5 to page 5, line 4
loading a global wake-up variable signalling process additions of processes to said process list into their associated cache memory unit (2),	Fig. 3, S20	Page 4, line 5 to page 5, line 4
for switching into a low-power mode if said process list contains no process for	Fig. 3, S26	Page 4, line 32 to page 5, line 4

execution by said processors (1)		
and for switching into a normal-power mode if said wake-up variable signals an addition of a process to said process list.	Fig. 3, S26	Page 5, lines 5-12

The following analysis of independent claim 6 is presented for convenience:

Element	Figure(s)	Paragraph(s) and/or page(s)
6. Processor for use in a multi-processor computer system comprising		
at least two processors (1) for parallel execution of processes,	Fig. 5, 1	Page 3, line 31 to page 4, line 4; page 5, lines 19-27.
at least two cache memory units (2), each being associated with and connected to a separate processor (1),	Fig. 5, 2	Page 3, line 31 to page 4, line 4; page 5, lines 19-27.
a connection bus (4) connecting said processors (1) and said cache memory units (2), and	Fig. 5, 4	Page 3, line 31 to page 4, line 4; page 5, lines 19-27.
a process list unit (3) connected to said connection line (4) for storing a process list of processes to be available for execution by said processors (1), wherein said processors (1) are adapted for	Fig. 5, 3	Page 4, line 5 to page 5, line 4
loading a global wake-up variable signalling process additions of processes to said process list into their associated cache memory unit (2),	Fig. 3, S20	Page 4, line 5 to page 5, line 4
for switching into a low-power mode if said process	Fig. 3, S26	Page 4, line 32 to page 5, line 4

list contains no process for execution by said processors (1)		
and for switching into a normal-power mode if said wake-up variable signals an addition of a process to said process list.	Fig. 3, S26	Page 5, lines 5-12

The following analysis of independent claim 7 is presented for convenience:

Element	Figure(s)	Paragraph(s) and/or page(s)
7. Method of scheduling the execution of processes in a multi-processor computer system comprising		
at least two processors (1) for parallel execution of processes,	Fig. 5, 1	Page 3, line 31 to page 4, line 4; page 5, lines 19-27.
at least two cache memory units (2), each being associated with and connected to a separate processor (1),	Fig. 5, 2	Page 3, line 31 to page 4, line 4; page 5, lines 19-27.
a connection bus (4) connecting said processors (1) and said cache memory units (2), and	Fig. 5, 4	Page 3, line 31 to page 4, line 4; page 5, lines 19-27.
a process list unit (3) connected to said connection line (4) for storing a process list of processes to be available for execution by said processors (1), said method comprising the steps of:	Fig. 5, 3	Page 4, line 5 to page 5, line 4
loading a global wake-up variable signalling process additions of processes to said process list into their associated cache memory unit (2).	Fig. 3, S20	Page 4, line 5 to page 5, line 4
adding a process to said	Fig. 4, S31	Page 5, lines 5-12; page 5,

process list, and		line 28 to page 6, line 8.
changing the wake-up variable signalling said addition of a process to said process list thus causing said processor (1) to switch from a low-power mode into a normal-power mode.	Fig. 4, S33	Page 5, lines 5-12; page 5, line 28 to page 6, line 8.

The following analysis of independent claim 8 is presented for convenience:

Element	Figure(s)	Paragraph(s) and/or page(s)
8. Method of scheduling the execution of processes in a multi-processor computer system comprising		
at least two processors (1) for parallel execution of processes,	Fig. 5, 1	Page 3, line 31 to page 4, line 4; page 5, lines 19-27.
at least two cache memory units (2), each being associated with and connected to a separate processor (1),	Fig. 5, 2	Page 3, line 31 to page 4, line 4; page 5, lines 19-27.
a connection bus (4) connecting said processors (1) and said cache memory units (2), and	Fig. 5, 4	Page 3, line 31 to page 4, line 4; page 5, lines 19-27.
a process list unit (3) connected to said connection line (4) for storing a process list of processes to be available for execution by said processors (1), said method comprising the steps of:	Fig. 5, 3	Page 4, line 5 to page 5, line 4
loading a global wake-up variable signalling process additions of processes to said process list into their associated cache memory unit (2),	Fig. 3, S20	Page 4, line 5 to page 5, line 4
switching into a low-power	Fig. 3, S26	Page 4, line 32 to page 5,

mode if said process list contains no process for execution by said processor (1)		line 4
switching into a normal-power mode if said wake-up variable signals an addition of a process to said process list, and	Fig. 3, S26	Page 5, lines 5-12
accessing said process list to get said added process for execution.	Fig. 3, S23	Page 4, lines 19-29

VI. GROUNDS of REJECTION to be REVIEWED ON APPEAL

The issues in the present matter are whether:

1. under 35 USC 102(a), claims 1-4 and 6-10 are unpatentable over Engel.
2. under 35 USC 10(a), claim 5 is unpatentable over Engel in view of Culler.

The rejections under 35 USC 112 (claims 1-10) and 35 USC 101 (claim 10) are not appealed.

VII. ARGUMENT

I. Rejection of Claims 1-4 and 6-10 as Unpatentable Over Engel

The rejection of claim 1 is illustrative. It states in part:

Engel discloses a multi-processor computer system comprising at least two processors (1) for parallel execution of processes (*See abstract: An alternative embodiment details the peripheral device be an unit processor in a multiprocessor system*), a connection bus (4) connecting said processors (1) and said cache memory units (2) (*See figure 6: A system power controller microcontroller bus 610 is available for connecting the processors*), and a process list unit (3) connected to said connection line (4) for storing a process list of processes to be available for execution by said processors (1) (*See figure 6: Instructions in memory are able to be passed through bus 610*), wherein said processors (1) are adapted for loading a global wake-up variable signalling process additions of processes to said process list into their associated cache memory unit (2) (*See column 7, lines 43-45: A subsystem power controller is used*), for switching into a low-power mode if said process list contains no process for execution by said processors (1) (*See column 14, lines 66-68: A Power Down signal is given and turns off specified parts of a system*) and for switching into a normal-power mode if said wake-up variable signals an addition of a process to said process list (*See column 14, lines 39-42: A Power Up signal is given and turns on specified parts of the system*).

The rejection further states:

Engel has not specifically taught at least two cache memory units, each being associated with and connected to a separate processor. However, caches are well known to speed up the retrieval of data so that data does not need to be retrieved in a slow manner by going to the main memory off chip. [I]t would have obvious...to have the computer system of Engel comprise at least two cache memory units, each being associated with and connected to a separate processor, for the desirable purpose of speeding up the retrieval of data.

The correspondence drawn between the claimed features of the invention and the feature disclosed by Engel is flawed.

Essentially, Engel teaches centralized subsystem power control responsive to

commands from command sources to power up or power down various peripheral control units, e.g., by controlling relay switches.

Figure 18 of Engel illustrates various possible values of an SPC (Subsystem Power Controller) function byte, value 01 being power up and value 02 being power down. As illustrated in Figure 3 of Engel, the SPC controls power to a group of peripheral devices controlled by a peripheral control unit. The SPC is responsive to a "command source," i.e., an entity that initiates a change in power status. As illustrated in Figure 6, the SPC may be microcontroller based and may use redundant processors.

In the invention, processors control their own power states in response to the value of a cached global variable. In Engel, a power controller explicitly controls the power states of peripheral devices via relays (Engel, Figure 14).

The rejection seems to imply that the peripheral devices of Engel might in fact be processors. Even if this were so, explicitly controlling the power states of processors via relays does not meet the features of the present invention as claimed. There is no mention in Engel of a process list. Simply stating that "Instructions in memory are able to be passed through bus 610" does not meet this feature of the claims. There is no mention in Engel of a global wake-up variable signalling process additions of processes to the process list. Simply stating that "A subsystem power controller is used" does not meet this feature of the claims.

The rejection finds caches "inherent" in the system of Engel. Even if this were so, explicitly controlling via relays the power state of processors equipped with caches does not meet the features of the present invention as claimed.

Hence it may be seen that in numerous respects, Engel fails to anticipate the invention of claim 1.

The same features of claim 1 absent from Engel are also recited in independent claim 6, and corresponding features are recited in independent claim 8. For the same reasons, Engel therefore fails to anticipate the inventions of claim 6 and 8.

With respect to claim 7, the rejection states summarily, "As per claim 7, Engel discloses the limitations for reasoning similar to that of claims 1 & 2." That reasoning is flawed for the same reasons as that of the rejection of claims 1 and 2.

Furthermore, claim 7 recites in part "adding a process to said process list, and changing the wake-up variable signalling said addition of a process to said process list thus causing said processor (1) to switch from a low-power mode into a normal-power mode." No such features are taught or suggested by Engel.

With regard to dependent claims 2-4 and 10, these claims depend from independent claims 1 and 7, which have been shown to be patently distinguishable over the cited reference. Accordingly, these claims are also patently distinguishable and allowable over the cited references by virtue of their dependency upon an allowable base claim.

II. Rejection of Claim 5 as Unpatentable Over Engel in View of Culler

The flaws in the rejection of claims 1-4, 6-8 and 10 above are only magnified in the rejection of claim 5. The rejection points to the redundant processors of Figure 6 of Engel and states that it would have been obvious to use an invalidation based cache coherence protocol with respect to these processors.

Firstly, it is not at all clear that the processors of Figure 6 should have caches. Figure 6 shows a microcontroller implementation of a subsystem power controller. Microcontrollers commonly do not have caches.

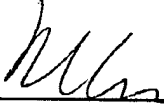
Secondly, even if the microcontrollers were to have caches and were to implement an invalidation based cache coherence protocol, it is other "processors" (peripheral devices) in Engel, other than the processors of the subsystem power controller, that have their power states controlled. They are controlled explicitly through relays in response to explicit commands, not through the mechanism of a cache coherence protocol as in claim 5. Claim 5 therefore would not have been obvious given Engel and Culler.

In view of the above, applicant submits that all of the above referred-to claims are patentable over the teachings of the cited references.

VIII. CONCLUSION

In view of the above analysis, it is respectfully submitted that the referenced teachings, whether taken individually or in combination, fail to anticipate or render obvious the subject matter of any of the present claims. Therefore, reversal of all outstanding grounds of rejection is respectfully solicited.

Date: 5/13/08


By: Michael Ure
Attorney for Applicant
Registration No. 33,089

IX. APPENDIX: THE CLAIMS ON APPEAL

1. Multi-processor computer system comprising at least two processors (1) for parallel execution of processes, at least two cache memory units (2), each being associated with and connected to a separate processor (1), a connection bus (4) connecting said processors (1) and said cache memory units (2), and a process list unit (3) connected to said connection line (4) for storing a process list of processes to be available for execution by said processors (1), wherein said processors (1) are adapted for loading a global wake-up variable signalling process additions of processes to said process list into their associated cache memory unit (2), for switching into a low-power mode if said process list contains no process for execution by said processors (1) and for switching into a normal-power mode if said wake-up variable signals an addition of a process to said process list.
2. Multi-processor computer system as claimed in claim 1, wherein said processors (1) are adapted to switch into the normal-power mode if the wake-up variable held in the associated cache memory units (2) is changed due to an addition of a process to said process list.
3. Multi-processor computer system as claimed in claim 1, wherein said processors (1) are adapted to execute a store command on the wake-up variable when adding a process to said process list.

4. Multi-processor computer system as claimed in claim 1, wherein said processors (1) are adapted to send a request to other processors to drop the wake-up variable from their associated cache memory unit when adding a process to said process list.
5. Multi-processor computer system as claimed in claim 1, wherein said computer system is adapted for implementing an invalidation based cache coherence protocol.
6. Processor for use in a multi-processor computer system comprising at least two processors (1) for parallel execution of processes, at least two cache memory units (2), each being associated with and connected to a separate processor (1), a connection bus (4) connecting said processors (1) and said cache memory units (2), and a process list unit (3) connected to said connection line (4) for storing a process list of processes to be available for execution by said processors (1), wherein said processor (1) is adapted for loading a global wake-up variable signalling process additions of processes to said process list into its associated cache memory unit (2), for switching into a low-power mode if said process list contains no process for execution by said processor and for switching into a normal-power mode if said wake-up variable signals an addition of a process to said process list.
7. Method of scheduling the execution of processes in a multi-processor computer system comprising at least two processors (1) for parallel execution of processes, at least two cache memory units (2), each being associated with and connected to a separate processor (1), a connection bus (4) connecting said processors (1) and said cache memory units (2),

and a process list unit (3) connected to said connection line (4) for storing a process list of processes to be available for execution by said processors (1), said method comprising the steps of: loading a global wake-up variable signalling process additions of processes to said process list by a processor (1) into its associated cache memory unit (2), adding a process to said process list, and changing the wake-up variable signalling said addition of a process to said process list thus causing said processor (1) to switch from a low-power mode into a normal-power mode.

8. Method of executing a process by a processor in a multi-processor computer system comprising at least two processors (1) for parallel execution of processes, at least two cache memory units (2), each being associated with and connected to a separate processor (1), a connection bus (4) connecting said processors (1) and said cache memory units (2), and a process list unit (3) connected to said connection line (4) for storing a process list of processes to be available for execution by said processors (1), said method comprising the steps of: loading a global wake-up variable signalling process additions of processes to said process list into an associated cache memory unit (2), switching into a low-power mode if said process list contains no process for execution by said processor (1), switching into a normal-power mode if said wake-up variable signals an addition of a process to said process list, and accessing said process list to get said added process for execution.

10. Computer program comprising computer program code means for causing a computer

to perform the steps of the method as claimed in claim 7 if said methods are executed by
said computer.

X. APPENDIX: RELATED PROCEEDINGS

NONE

XI. APPENDIX: EVIDENCE

NONE